Experiments with Cisco MIBs, QoS, and SNMP Features For Adapting SAAM Flow/Path Based Routing and Control

(SAAM Proxy-Box)

By Cary Colwell

Research and Tools CD Accompanying this Report:
A **CD accompanies this report**, containing **downloaded Cisco MIBs, Cisco QoS Related PDF documents, Adventnet's SNMP development kit and Mib Browser, test router configuration files, and some router development tool programs**. The CD provides a valuable springboard for any follow-up research. Many of my assessments about Cisco's levels of QoS support are linked to references I've gathered and bundled on the CD. The CD also contains **project reports for previous related work performed by SAAM researchers and Advanced Network Programming students** at Naval Postgraduate School.

Background:
Cisco IOS (Internetworking Operating System) supports three end-to-end QoS models: Best Effort, Integrated and Differentiated Services.
(See report CD:docs\qcintro.pdf.). IOS includes RSVP support for signaling for and obtaining bandwidth, delay, and other network resource.
RSVP is a protocol used by hosts to request, reserve and obtain QoS from a network of Cisco routers. RSVP is tightly associated with other QoS mechanisms including Integrated Services (Int-Serv) and Differentiated Services. Int-Serv is a QoS framework which relies on an underlying resource reservation mechanism. Cisco IOS uses RSVP to provide that underpinning. It's no accident, then, that exploring Cisco's support of RSVP has yielded up ways to access and possibly control the flow/path-based routing data a Cisco router can make available via an open, **network management** (**NM**) interface such as the routers own management information base (mib) tree and SNMP agent.

SNMP protocol is used by Cisco router agents and NM hosts to allow management monitoring and limited control. We have obtained most current Cisco RSVP and IntServ Mibs and have demonstrated RSVP and IntServ tables are indeed enabled and updated by the IOS. To do this, we had to perform experiments using two QoS enabled Cisco routers, since the information we located on the matter was limited and spread across many sources. Cisco QoS is undergoing development even now, and details of IOS internals very limited, but we were able to find affirming answers from our own experiments.

We require SNMP agent and mib accessibility to Cisco router QoS variables which either directly mimic or map to SAAM path/flow-based routing table data. To-date, we have identified traditional and QoS Cisco Mibs (and their objects) of interest:

RFC1213-MIB (Interface, Link and Network Layer (ARP and IP routing tables)
RSVP-MIB (rsvp Session, Sender, ResvTable, ResvFwdTable, IfTable, NbrTable)
INT-SERV-MIB (intSrv IfAttribTable, FlowTable)

We have uncovered some other flow-related Mibs, which haven't been explored significantly, as appear to be geared more towards Cisco NM products rather than

IOS routers. However, I'll note them here for possible future research:
DOCS-QOS-MIB (requires DOCS-IF-MIB to be preloaded)
All of these mibs are available on the report CD.

Discussion:
We conducted research and experiments to better understand those aspects of the Cisco IOS which must be configured and/or controlled by SAAM Proxy SNMP Agent units lashed to Cisco routers. We attempted to discover which elements exposed by Cisco's MIB trees best afford Flow/Path-based routing information.

We performed a survey of the Cisco QoS mib sources and singled out the RSVP-MIB.my and INT-SERV-MIB.my files as containing the path and flow routing data structures which map to SAAM routing.
More fundamentally, we attempted to discover if Cisco IOS Routers SNMP agent accesses/updates/ or otherwise uses the RSVP and INT-SERV mibs.

No body, even contacts at Cisco, could tell us for sure. We set out to see if we could experimentally answer this using two 2611 Cisco Access Routers, each with IOSv12.0, and using an SNMP Mib Browser from Adventnet, loaded with Mibs obtained from the Cisco public mib ftp site: **ftp://ftp.cisco.com/pub/mibs/v2**.
and later using updated mibs obtained from Fred Baker- Cisco/IETF and staff.

The figure below depicts the basic test topology used to conduct these experiments.

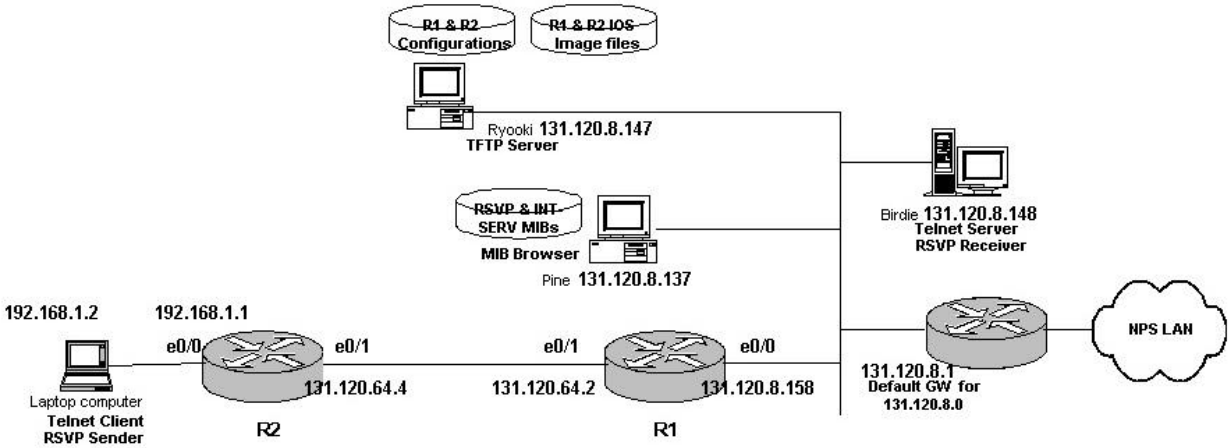

Figure 1. Experimental test network comprised of RSVP and SNMP enabled routers.

Router configuration tools were loaded on host Ryooki in the form of a tftp server and terminal emulator program (this one is superior to HyperTerminal if you need to upgrade or replace the IOS, wipe out the IOS or otherwise cause the router to break, or need to perform any extraordinary administration or configuration beyond the capabilities of the IOS command line interpreter. HyperTerminal cannot emit a "break" which is required to get into the router monitor program. The tftp server allows installation, backup, and restoral of IOS images, router configuration files, and bootstrap programs. A lap top computer runs a telnet client program to simulate a best-effort

flow source, and host birdie was installed with a telnet server to act as a complementary flow destination. Neither flow endpoint was operated in RSVP-enabled mode. The laptop, running Windows XP is capable of running RSVP, but was not configured to do so. The default gateway is the NPS LAN's gateway router to our lab and provides Internet, DHCP, and DNS service to Birdie and Ryooki. Host pine was equipped with Adventnet's MIB Browser package to perform SNMP queries on the test routers. Router R1 is the left-most cylinder in Figure 1. Router R2 is the center-most Cylinder. Together, they form the test network for subnets 192.168.0.0 and 131.120.64.0. More on the test router configurations follow later. Each router is configured via serial connection through its console control port. Asynchronous serial settings are 9600, no parity, 8 data bits, 1 stop-bit, and are used to configure the controlling terminal emulator (term.exe or HyperTerminal). Terminals are run in separate windows operating on the laptop and Ryooki. The serial connections are not shown in figure 1.

Early testing with the official Cisco RSVP and INT-SERV mibs proved frustrating as the mibs downloaded from the Cisco ftp and other sites were out of date and could not compile no matter how the mibs and browser were configured. Many efforts were made to establish the state of the router agents and the mibs. The first successful effort was obtaining a pair of mibs from a contact at Adventnet, who merely culled out the mib sources directly from the IETF RFC web site. These mibs compiled OK, and revealed slight population of RSVP and INT-SERV variables and tables during router probing, we could be wasting more time trying to activate non-supported objects if these mibs did not follow the design directions Cisco was pursuing. The author of the mibs was Mr. Ed Baker of Cisco, and his email address was given in the Mib source, so we contacted him directly.
Emails to Mr. Baker brought us more up-to-date mib files and an apologetic "whoops". Our inquiry apparently has helped them get their house in order. The **new mibs** compiled and loaded easily. Extensive testing using these mibs with our test network confirmed their utility. These mibs are included on the report CD.

Since SAAM mainly concerns itself with flow/path-based routing, we were particularly interested in the object Identifiers (OIDs) specified in mib-2.intSrv.intSrvObjects.intSrvFlowTable.

We used the test configuration illustrated in **figure 1**. The following describes the procedures and router configuration commands used to conduct the experiments.

Two routers were configured to form a small private test bed, designed to be non-intrusive to the remainder of the LAN.
RIPv2 was **configured** as the **IP routing algorithm**, since minimum configuration is required. The following commands are used (all of the configuration commands must be issued in enable mode):

```
router rip
version 2
network 131.120.0.0
network  192.168.1.0
```

In this stub test topology, no **static route configuration** is necessary for **routers**. Routers will each show two routes due to their directly connected networks, one for each interface. Routing

tables should each contain an indirect route: R1 should show a RIP-advertised route to 192.168.1.0 and R2 should show a RIP-advertised route to 131.120.0.0. Verify this using the command:

show ip route

Dump of **IP routing tables for R1 and R2** under quiescent (no snmp or administrative route table manipulations):

Router 1:

```
    131.120.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       131.120.64.0/27 is directly connected, Ethernet0/1
C       131.120.8.0/22 is directly connected, Ethernet0/0
R    192.168.1.0/24 [120/1] via 131.120.64.4, 00:00:21, Ethernet0/1
```

Router2:

```
    131.120.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       131.120.64.0/27 is directly connected, Ethernet0/1
R       131.120.8.0/22 [120/1] via 131.120.64.2, 00:00:07, Ethernet0/1
    192.168.1.0/30 is subnetted, 1 subnets
C       192.168.1.0 is directly connected, Ethernet0/0
```

However, **hosts** (flow generators, flow sinks, snmp management stations, etc.) requiring two-way connectivity (as in pinging) **must be configured with a gateway** route to the test router subnets, since the NPS Lab network routers and servers know nothing about the test router topology. If this is not done, replies from the NPS lab hosts will be directed to 131.120.8.1, the default gateway configured by DHCP. Once all participating hosts have static routes for replying to the test networks, pings from these hosts will get their reply packets. Otherwise they may be directed to 131.120.8.1 and you will get ICMP redirect messages.

SNMP was configured as follows:
Routers were given private and public SNMP community strings and the SNMP agents were enabled. Test hosts were configured to establish test flows from end-to-end. A telent server was set up on host Birdie. This was used as the RSVP receiver. A telnet client was run on a laptop at the stub-end of the test network, serving as the RSVP sender. Static RSVP PATH and RESV conditions were established by using the commands:

```
snmp-server engineID local 0000000902000002B93EC860
snmp-server community public RO
snmp-server community saamReadWrite RW
snmp-server location lab
```

(see Configuring SNMP reference in fcd301.pdf on the report CD)

RSVP for QoS was configured as follows:

IP rsvp sender 131.120.8.148 192.168.1.2 tcp 0 0 192.168.1.1 Ethernet0/0 3400 7500

<static path>

This provides data fields in the RSVP static/simulated path message covering the sender's and receiver's address, session protocol, wild carded sender and receiver ports, address and interface of next-hop router closest to sender, average and burst data rates in kbps.

IP rsvp reservation 131.120.8.148 192.168.1.2 tcp 0 0 192.168.1.1 Ethernet0/0 SE load 3400 3750

<static RESV message>

This provides data fields in the RSVP static/simulated RESV message.
(For details on both commands, see QoS Commands for IOS 12.0 in qrcmda.pdf on the report CD)

Details **on how to use IOS command line interface** commands in-general have been covered extensively in previous reports and projects, however a definitive references are available on-line and on the report CD.
(see IOS Command reference for basic configuration instructions in qcducing.pdf on the report CD)

Here's a synopsis of **what the static commands are simulating:**
RSVP enabled sender hosts begin the process by sending an RSVP PATH message into the network. RSVP on the ingress router propagates the PATH message, apparently working with the resident routing protocol (RIP, OSPF, etc) on to the next hop router, according to its routing information. PATH message is routed to the receiver host which is specified in the PATH message. No resource reservations are made yet. To keep things initially simple, I used a single routing protocol (RipV2). Since the NPS LAN routers are not accepting advertisements from Rip, we were afforded isolation, and didn't need to worry about advertising our routes to the base. It would be interesting to observe what path is used if different, multiple routing algorithm processes are run simultaneously and they each favor different paths.
The RSVP enabled receiver host receives the PATH message and waits till it wants to communicate with sender. Receiver sends an RSVP RESV message back into the network along the same path traversed by the sender's PATH message.

Hosts and routers use RSVP to deliver QoS requests to the routers along the paths of the data stream and to maintain router and host state to provide the requested service, usually bandwidth and latency. RSVP uses a mean data rate--the largest amount of data the router will keep in the queue--and minimum QoS (that is, guarantee of the requested bandwidth specified when you made the reservation using RSVP) to determine bandwidth reservation.

A host uses RSVP to request a specific QoS service from the network on behalf of an application data stream. RSVP requests the particular QoS, but it is up to the interface queuing mechanism to implement the reservation. RSVP carries the request through the network, visiting each node the network uses to carry the stream. At each node, RSVP attempts to make a resource reservation for the stream using its own admission control module, exclusive to RSVP, which determines whether the node has sufficient available resources to supply the requested QoS.

An application could send traffic at a rate higher than the requested QoS, but the application is guaranteed only the minimum requested rate. If bandwidth is available, traffic surpassing the requested rate will go through if sent; if bandwidth is not available, the exceeding traffic will be dropped. If the required resources are available and the user is granted administrative access, the RSVP daemon sets arguments in the packet classifier and packet scheduler to obtain the desired QoS. The classifier determines the QoS class for each packet and the scheduler orders packet transmission to achieve the promised QoS for each stream. If either resource is unavailable or the user is denied administrative permission, the RSVP program returns an error notification to the application process that originated the request.

WFQ or WRED sets up the packet classification and the scheduling required for the reserved flows. Using WFQ, RSVP can deliver an integrated services Guaranteed Rate Service. Using WRED, it can deliver a Controlled Load Service.

IOS Commands to configure WFQ used on each interface of the router:

```
fair-queue 1200 256 234
ip rsvp bandwidth 7500 7500
```

Using Debug Dumps to Verify and Analyze Events, Packets, Router State, Performance

IOS provides a debug facility which can be enabled for broad or narrow sets of router operations. Debugging can be very CPU intensive, and it is possible to render a busy router unusable, even to the point of ignoring serial console commands. Discussion of all important aspects of this facility are beyond the scope of this report, but command references and facility description are in the PDF document section of the accompanying CD. To verify operation of RSVP, aid in configuration and verification of RSVP, use the command: **debug ip rsvp.**
As with most IOS commands, there are event and packet related sub-categories to broaden or narrow the scope of your analysis.
In an **emergency, disable all possible debugging** options with one command: **no debug all.** You may type this command "in the blind" in the event of an overloaded router, in order to save lost work from a power reset.

SNMP MIB and Browser Installation and Experiments:

I downloaded and installed Adventnet's SNMP utilities (also on the report CD) to use the MIB Browser used by CS4552 students in previous research. Simply configuring the target router's Ipv4 address or host name (if name server is reachable) and getting IP routing configured on the routers, the browser is able to load/compile mibs, engage the router with GET, PUT, and other SNMP operators to display and possibly modify OIDs in the selected MIB tree.

Procedure for installing Cisco QoS MIBS:
When you first start the mibBrowser, the only mib loaded is **RFC1213-MIB**.
The default loaded MIB is loaded from the adventnet directory:
program files\Adventnet\SNMPUtilities\mibs
This mib gives you the standard suite of management variables, including the **IP routing table**.
To access the QoS variables on the Cisco routers, one must load the appropriate mibs.
These are RSVP-MIB and INT-SERV-MIB.

I have added downloaded Cisco MIBS from the ftp website cited earlier. There are dependencies the MIB compiler must be able to resolve which require loading some additional MIBs prior to loading the QoS MIBS: Loading IF-MIB from Cisco's public ftp site auto-loads Cisco's SNMPV2-MIB, which are required to load RSVP-MIB, which is required to load INT-SERV-MIB. The **procedure to load** is as follows:
File|Load MIB|Browse opens a directory browsing applet. Locate the Cisco Mib directory : **program files\Adventnet\SNMPUtilities\CiscoMibsCary\v2\IF-MIB.my**.
Click OK to compile and load.
I have created a directory above the Adventnet mib directory, containing the working Cisco MIBS as described previously. File|Load MIB|Browse opens a directory browsing applet. Locate the QoS MIB file, RSVP-MIB.my in
program files\Adventnet\SNMPUtilities\CiscoMibsCary\Modified Cisco\From Cisco .
Highlight RSVP-MIB.my and click open. Click OK to compile and load.
Repeat procedure with INT-SERV-MIB.my.

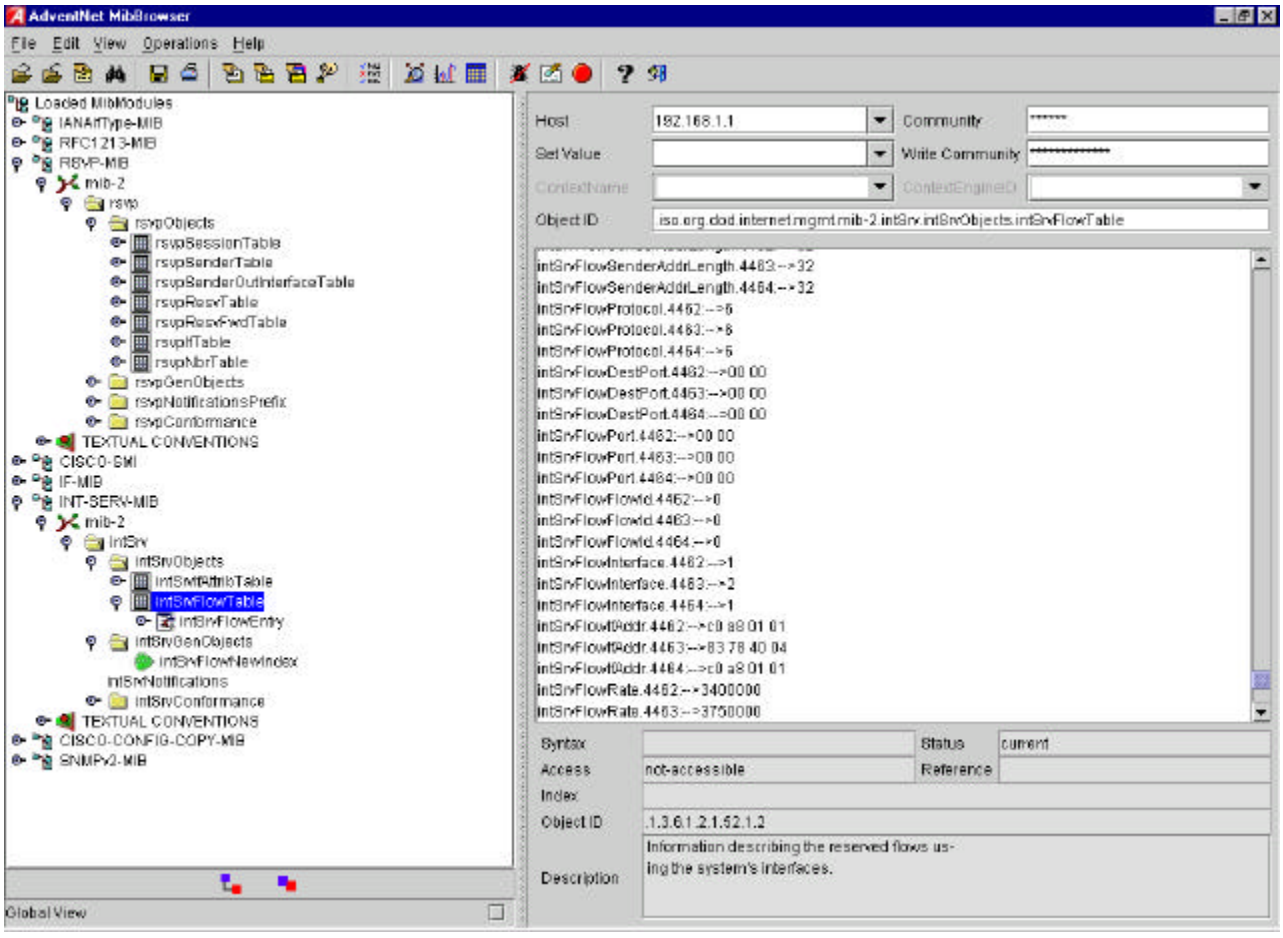

Figure 2. Adventnet's Mib Browser gui showing Cisco target mibs loaded.

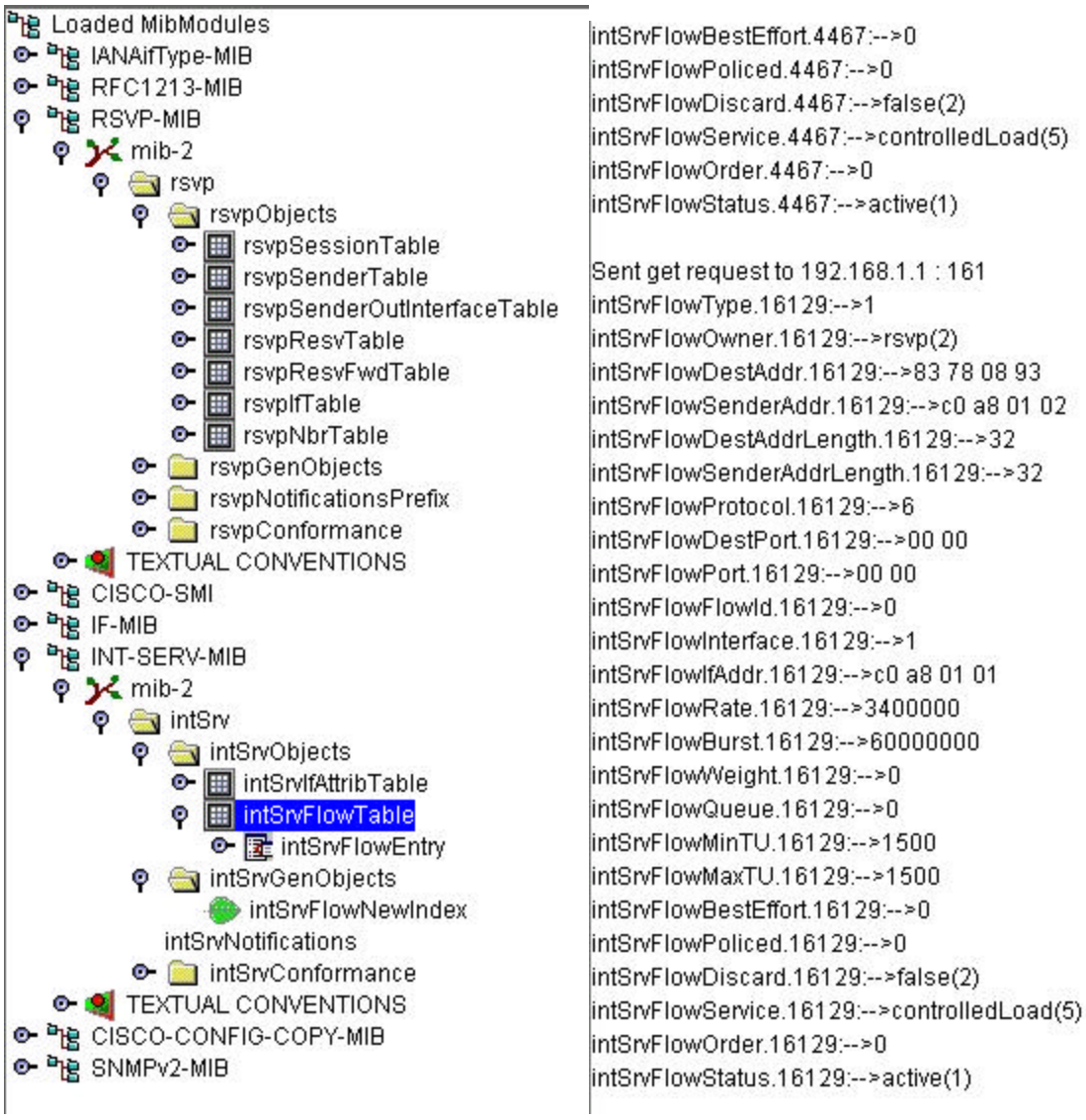

Figure 3. Close up of Mib trees and Result window after a dump of the entire intSrvFlowTable object.

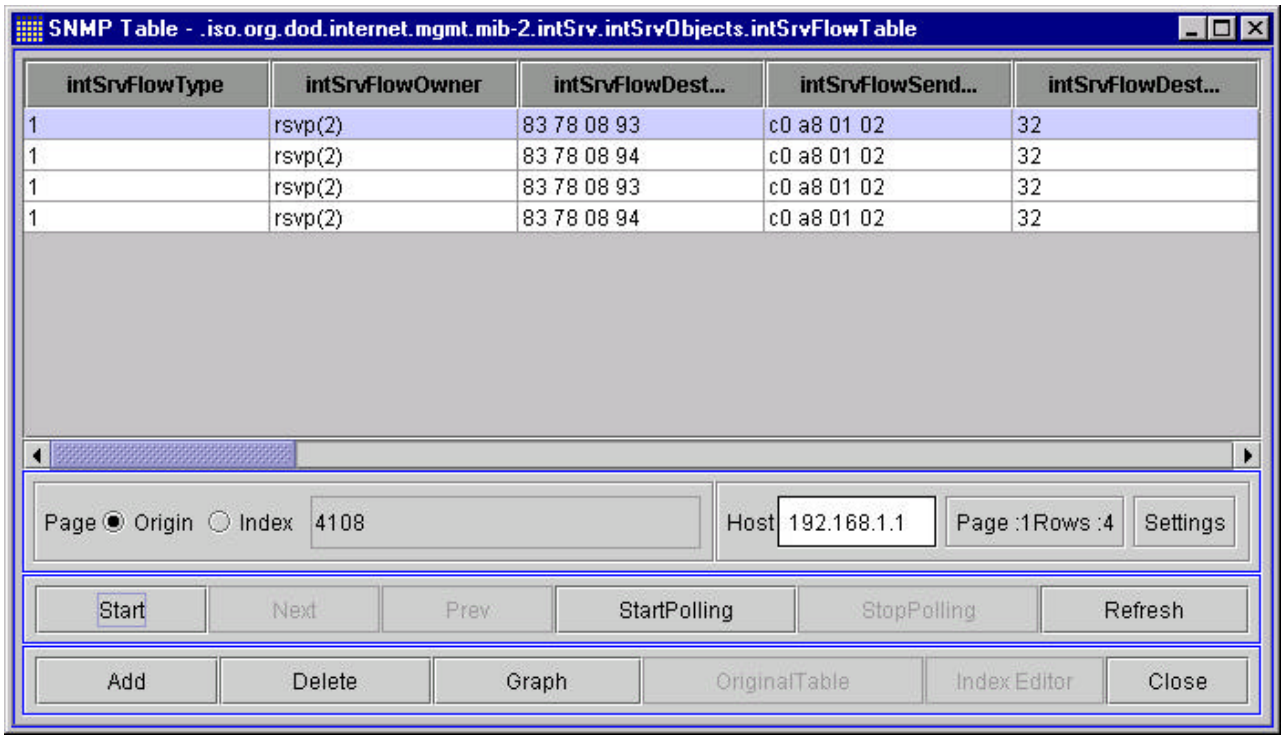

Figure 4. Example of MIB Browser's Table representation of the intSrvFlowTable. Example shows Best-Effort sessions established to characterize the data representation. All session flowIDs are zero (must move slider bar to see other columns in the table).

Though we succeeded in verifying that the router's agent does have hooks into these desired data structures, we noticed that there were some other INT-SERV and RSVP- mib objects not being updated. By further configuring QoS queuing disciplines, we were able to get most of those to come alive as well. Unfortunately, there is no centralized location for procedures or information on how to test any object in the MIB tree except for those which are common for network management (e.g. RFC1213-MIB, which contains ARP and Ipv4 routing tables). Thus, the need for these experiments.

By default, IOS uses FIFO-queue discipline for all interfaces unless configured otherwise. FIFO does not allow RSVP to allocate resources as evidenced by the following debug dump:

22:11:50: RSVP session 131.120.8.148_0: Sending **PATH** message for 131.120.8.148 on interface Ethernet0/1
22:11:50: RSVP session 131.120.8.148_0: IP to 131.120.8.148 length=172 checksum=5E69 (Ethernet0/1)
22:11:51: RSVP 131.120.64.2_4098-131.120.64.4_35373: message received from 131.120.64.2
22:11:51: RSVP 192.168.1.2_0-131.120.8.148_0: **RESV** message for 131.120.8.148 (Ethernet0/1) from 131.120.64.2
22:11:51: RSVP-TC: Attempting to remove QoS for reservation
22:11:51: RSVP-TC: Deleting conversation
22:11:51: Released conversation #1650 for 131.120.8.148, port 0
22:11:51: RSVP-TC: **Attempting to install QoS for session**

22:11:51: add_conversation - creating conversation 80CEC9AC. rsb=80EF58C0
22:11:51: RSVP-TC: **Assigning NO QoS to reservation**
22:11:57: RSVP 192.168.1.1_4097-131.120.8.148_0: message received from 192.168.1.1
22:11:57: RSVP 192.168.1.2_0-131.120.8.148_0: Received PATH Message for 131.120.8.148(Ethernet0/0) from 192.168.1.1, rcv IP ttl=254

By configuring Weighted-Fair Queuing for each interface, debug dumps reveal that RSVP not only succeeds in assigning QoS to the test reservation, but the SNMP get probe shows that corresponding entries in RSVP and INT-SERV flow tables become populated with corresponding data.

A sample debug session dump showing router2 processing of **RSVP PATH and RESV messages** as a result of static reservation steps above:

8w1d: RSVP: version:1 flags:0000 type:**PATH** cksum:0000 ttl:255 reserved:0 len1) from 131.120.64.2
8w1d: RSVP session 131.120.8.148_0:
8w1d: TIME_VALUES type 1 length 8 : 00007530
8w1d: FLOWSPEC type 2 length 48:
8w1d: version = 0 length in words = 10
8w1d: service id = 2, service length = 9
8w1d: tspec parameter id = 127, tspec flags = 0, tspec length = 5
8w1d: average rate = 468750 bytes/sec, burst depth = 7500000 bytes
8w1d: peak rate = 468750 bytes/sec
8w1d: min unit = 0 bytes, max unit = 1500 bytes
8w1d: rspec parameter id=130, rspec flags=0, rspec length=2
8w1d: requested rate=468750, slack=0
8w1d: FILTER_SPEC type 1 length 12:
8w1d: Source 192.168.1.2, udp_source_port 0
8w1d: SENDER_TEMPLATE type 1 length 12:, burst depth=7500000 bytes
8w1d: Source 192.168.1.2, udp_source_port 0
8w1d: peak rate =1250000 bytes/s
8w1d: SENDER_TSPEC type 2 length 36:w1d: min unit=0 bytes, max unit=1500 bytes
8w1d: ADSPEC
8w1d: Token bucket fragment (service_id=1, length0 is now available
8w1d: ADSPEC type 2 length 84
8w1d: version=0 length in words=19 Path MTU:-1
8w1d: General Parameters break bit=0 service length=8 Guaranteed Service break bit=0 service length=8
8w1d: IS Hops:0 Path Delay (microseconds):0
8w1d: Minimum Path Bandwidth (bytes/sec):2147483647 Path Jitter (microseconds):0
8w1d: Path Laten=0
8w1d: Path delay since shaping (microseconds):0
8wld: (Ethernet0/0) from 192.168.1.1, rcv IP ttl=254
8w1d: Path Jitter since shaping (microseconds):0
8w1d: RSVP: version:1 flags:0000 type:**RESV** cksum:0000

8w1d: Controlled Load Service break bit=0 service length=0RN to get started.-compression statistics
8w1d: RSVP 192.168.1.2_0-131.120.8.148_0: RESV message for 131.120.8.148 (Ethernet0/0) from 192.168.1.1
8w1d: RSVP session 131.120.8.148_0: Sending PATH message for 131.120.8.148 on interface Ethernet0/1
8w1d: RSVP: version:1 flags:0000 type:PATH cksum:40B4 ttl:254 reserved:0 length:172
8w1d: SESSION type 1 length 12:
8w1d: Destination 131.120.8.148, Protocol_Id 6, Don't Police , DstPort 0
8w1d: HOP type 1 length 12: 83784004seconds
8w1d: : 00000000
8w1d: TIME_VALUES type 1 length 8 : 00007530

Experiments to Change Tables and Scalars:

Investigating **ipRouteTable** from **RFC1213**:1.3.6.1.2.1.4.21,
I somehow deleted route on R1 for 131.120.8.158 so that snmp couldn't route and mibBrowser couldn't contact the router's agent. No responses. Does this suggest you can effectively control routing via SNMP? I'm not sure how this happened. I think I found a field in the ipRouteTable for **ipRouteType** which allows an entry of one of three values:
other(1), invalid(2), direct(3), indirect(4). For the 131.120.8.0 route, I tried changing this field and the router stopped routing. All ability to ping or use SNMP, telnet or do anything with the routers over the network was lost.
SETs on some variables allow you to perform simple operations like start/stop IP forwarding altogether. Operations like this can cut you off from the managed device entirely, with no hope to restore connectivity but to gain physical control of the device (reboot/reset/local serial console reconfiguration). But the ipRouteTable doesn't seem to allow changes at all. How I deleted the default route (I think that's the one since a console query, show ip route, shows the default route to 131.120.0.0 via 131.120.8.158 eth0/0 is gone, but **still in the running configuration**, confirmed from the serial console with the IOS CLI command: **show running-configuration**.
Puzzling! Rip should refresh the info in 30 seconds, but the default route never comes back!
The routing table for R1 should look like the following dump:
router1#
router1#sho ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is not set

131.120.0.0/16 is variably subnetted, 2 subnets, 2 masks ←===Means nothing
C 131.120.64.0/27 is directly connected, Ethernet0/1

C 131.120.8.0/22 is directly connected, Ethernet0/0 ←==**This is gone!**
R 192.168.1.0/24 [120/1] via 131.120.64.4, 00:00:01, Ethernet0/1

Without a specific route for the directly connected network 131.120.8.0, no snmp or any other IP traffic could flow, hence the **abrupt cutoff of response at the mibBrowser** machine.
As an experiment, I tried restarting RIP by first unconfiguring it, then reconfiguring RIP:

```
no  router rip
router rip
version 2
network 131.120.0.0
```

Merely reconfiguring the network does no good. I needed to kill the routing algorithm process to get things working again.
Result, the route for 131.120.8.0 came back. Something about the snmp didling may have messed up the routing algorithm process on Router1. SNMP probing was restored.
Further work should be performed to examine repeatability of this experiment.

Next, an attempt was made to add a row to the routing table. The Browser would not allow this operation. But it was soon noticed that the intServFlowTable did allow new row creation from the browser (at least the browser offered a menu selection which offered up a new row data entry gui). Analysis of table definition and element MIB definitions explains the variation in variable accessibility.

Importance of Understanding Scalar and Table Access Values, Modification, and Creation:
At first it seems there's a mystery as to why ipRouteTable has an access value of "not-accessible", yet the table is demonstrably readable. Similar confusion may easily follow access values for read, read-only, read-write, read-create, etc.
Answer: **Whole snmp tables** are objects (**definitions, descriptions** of how indexing is implemented, etc.), **not instances**. You **cannot** do a get or set on any table as a whole. Nor can you retrieve an entire row in one operation. Therefore, tables have access = not-accessible. Reads and writes to tables must be done on individual elements of the table. Access values are not "access rights", but denote maximum access (max-access) agents apply to data entities. GetBulk may be used to get large portions of a table but doesn't guarantee retrieval of all the elements of a table or data only for one table.
For SMIv2 the following ACCESS values/descriptions are useful to note:

*For LEAF-nodes only

Read-only: Not changeable by network manager *
Read-write: Net Manager may modify *
Read-create: If all column elements are this, new rows may be added *
Access-for-notify: *
Not-accessible: Info or definition- naming purpose only.

Note the accuracy of these statements in the following example MIB definition for a route table (This MIB is not from Cisco):

```
routeTable          OBJECT-TYPE
  SYNTAX            SEQUENCE OF RouteEntry
  MAX-ACCESS        not-accessible      --manager can't retrieve whole table at-a-time
  STATUS            current
  DESCRIPTION       "This entity's routing table"
  ::={NEW-MIB 3}

routeEntry          OBJECT-TYPE     --table row definition!
 SYNTAX             RouteEntry
 MAX-ACCESS         not-accessible      --manager can't retrieve whole row at-a-time
 STATUS             current
 DESCRIPTION        "A route to a particular destination"
 INDEX              {dest, policy}
 ::={routeTable 1}

RouteEntry::=           --Definition of new type for row
   SEQUENCE{
      dest  IpAddress,
      policy  INTEGER,
      next  IpAddress
}
            -- now define the elements.  Net Mgr can retrieve these individually
dest OBJECT-TYPE
  SYNTAX            IpAddress
  ACCESS            read-only    --for net mgr to add new rows, all must be read-create
  STATUS            current
  DESCRIPTION       "The address of a particular destination"
  ::={route-entry 1}

policy OBJECT-TYPE
 SYNTAX             INTEGER{
                      costs(1) – lowest delay
                      reliability(2) –highest reliability
                    }
 ACCESS             read-only   --for net mgr to add new rows, all must be read-create
 STATUS             current
 DESCRIPTION        "The routing policy to reach that destination"
 ::={route-entry 2}

next OBJECT-TYPE
 SYNTAX             IpAddress
 ACCESS             read-write
  STATUS            current
 DESCRIPTION        "Internet address of next hop"
 ::={route-entry 3}
```

For a NM system like SAAM, this means (at least as far as the Cisco IOS r.12.0) the router agent prohibits "outsiders" from adding/creating rows in the IpRouteTable. Modifications to some elements in EXISTING rows of that table may be permitted, others not. Some row elements are given read-only access, and others have read-write. No elements in that table have read-create access.

Further investigation in SMI-2 structure creation reveals that rules for table use involves more considerations when outsiders like SAAM might need to add a table row. Tables for which the agent is to support outsiders creating or adding new rows to a table must incorporate a ROW-STATUS column:
Change = modify or add rows.
Assume a management system (e.g. SAAM) wants to change a table row. Changes may be too big to fit in a single PDU or single operation, hence may need multiple operations. Until all changes are ready, other managers should not read or write (use) the incomplete row. Thus a new column called ROW STATUS is required. This is analogous to the locking required in multithreaded situations to prevent data corruption during accesses of a shared object by more than one process at a time, or in database systems where a table or record is simultaneously read a process while being updated by another.

ROW STATUS column Values:

ACTIVE	Other managers may read or write the row
NOT-READY	mgr can change, other mgrs may not read or write
DESTROY	Agent can throw it away

IF Manager adds a new row he writes the following values to row status column 2-ways:

createAndGo	Create row in one operation, all elements given at once.
createAndWait	update row an element at a time, then set row status to ACTIVE

IF Manger wants to delete a row, just manager just does a SET on row status element for the row to DESTROY.

If there's more to do after processing the MIB, then there needs to be additional code in the management system software.

SNMP is NOT a performance oriented environment. Results may sometimes take seconds or longer to occur. Another big design consideration!

Observations and Discoveries:
If senders segment is down (router interface into senders segment is down) or sender is un reachable from the sender's RSVP enabled router, the QoS mechanisms are aware.
RSVP Debug output shows that the condition is repeatedly sent up the reservation path that the sender is no longer reachable in the PATH message. Restoring the interface to an up condition causes the restoration to propagate to the RSVP chain as well.
Any IntSrvFlowTable entries produced under Ipv4 addressing (all experiments conducted) show flow ID = 0. This is true even if multiple static reservations and sessions were statically configured. This corresponds to Best-Effort. Non-zero, positive Ids are producible if further QoS

configuration is carried out. The OID descriptor for the FlowID field claims this is the flow ID that the sender is using if it is an Ipv6 session.
This is corroborated by descriptors for other field members such as intSrvFlowType, which claims this field represents the type of the session (IP4, IP6, IP6 with flow information, etc). See next section on recommended follow-up work.

The following OID break-outs may be used to further development of SAAM flow/path-based routing and control. Most of these descriptions may be viewed by editing the appropriate MIB source file.

RSVP Table Object Summaries:

RsvpSessionTable rows have the following fields:

rsvpSessionNumber; The number of session represented by the row.
rsvpSessionType; The type of session (IP4, IP6, IP6 with flow info, etc)
rsvpSessionDestAddr; Destination address used by all senders in this session.
rsvpSessionDestAddrLength; CIDR prefix length of session address, =32 for IP4 host and multicast address, =128 for IP6 addresses.
rsvpSessionProtocol; IP protocol used by this session.
rsvpSessionPort; UDP or TCP port used as destination port for all senders in session.
rsvpSessionSenders; Count of distinct senders currently known to be part of the session.
rsvpSessionReceivers; Number of reservations being requested of system for this session.
rsvpSessionRequests; Number of reservation requests router is sending upstream for this session.

RsvpSenderTable rows have the following fields:

RsvpSenderType;
RsvpSenderDestAddr;
RsvpSenderAddr;
RsvpSenderAddr;
Dozens more, including Tspec, Adspec Rate, Burst, Bandwidth, Latency objects

RsvpSenderOutInterfaceTable; List of outbound interfaces PATH msgs use.
RsvpResvTable; State info displayed by single receiver's RESV msg per single sender.
RsvpResvFwdTable; State info displayed upstream in RESV messages.
RsvpIfTable; RSVP-specific attributes of router's interfaces.
RsvpNbrTable; Neighbor's RSVP system info.

INTSERV Table Object Summaries:

IntSrvIfAttribTable rows have the following fields:

IntSrvIfAttribAllocatedBits; bits/sec currently allocated to reserved sessions on this interface.
IntsrvIfAttribMaxAllocatedBuffer; Buffer space required for simultaneous. Burst of all reserved flows on this interface.
IntSrvIfAtribFlows; Number of reserved flows active on interface. Flow can be created either from a reservation protocol (e.g. RSVP or ST-II) or via static configuration info.
IntSrvIfPropagationDelay; Propagation delay interface introduces over bit prop. delays.
IntSrvAttribStatus; equals 'active' on interfaces configured for RSVP.

IntSrvFlowTable rows have the following fields:

IntSrvFlowType; Session type (IP4, IP6, IP6 with flow info, etc).

IntSrvFlowOwner; Process which installed flow in queue policy database.

IntSrvFlowDestAddr;

IntSrvFlowSenderAddr;

IntSrvFlowDestAddrLength;

IntSrvFlowProtocol; IP protocol used by a session.

IntSrvFlowDestPort; UDP or TCP port# used as dest for all senders in this session. If IP in-use, specified by intServFwdProtocol, = 50(ESP) or 51(AH).

IntSrvFlowPort; UDP or TCP port# used as source port for this session's sender. If IP in-use, specified by intServFwdProtocol, then = 50(ESP) or 51(AH).

IntSrvFlowFlowId; The flow ID that this sender is using if this is an IP6 session.

IntSrvFlowAddr; IP addr of ifEntry with corresponding reservation. Mainly to support interfaces with multiple IP addresses.

IntSrvFlowRate; Sender's reserved data stream rate. If ControlledLoad service flow, rate is derived from the Tspec, (r). If a Guaranteed service flow, rate is derived from the Rspec clearing rate parameter, (R).

IntSrvFlowBurst; Size of largest expected burst-at-a-time from sender. If less than sender's advertised burst size, receiver is asking network to provide flow pacing beyond what would be provided under normal circumstances. Such pacing is at network's option.

IntSrvFlowWeight; Used to prioritize traffic. Interpretation is implementation-specific.

IntSrvFlowQueue; Number of the queue used by this traffic. Since queue identifier usage varies, this is implementation-specific.

IntSrvFlowMinTU; Minimum message size for this flow. Policing algorithm will treat smaller messages as though they are this size.

IntSrvFlowMaxTU; Maximum datagram size for this flow conforming to the traffic spec. Cannot exceed MTU of the interface.

IntSrvFlowBestEffort; Number of packets that were remanded to best effort service.

IntSrvFlowPoliced; Number of packets policed since inception of flow's service.

IntSrvFlowDiscard; True if flow is to incur loss when traffic is policed. If false, policed traffic is treated as best effort traffic.

IntSrvFlowService; QoS service being applied to this flow.

IntSrvFlowOrder; In event of ambiguity, order in which classifier should make its comparisons. The row with intSrvFlowOrder=0 is tried first, and comparisons proceeding order of increasing value. Non-serial implementations of the classifier should emulate this behavior.

IntSrvFlowStatus; 'Active' for all active flows. May be used to install static classifier information, delete classifier info, or authorize such.

SNMP table access and manipulation should be fully understood and due considerations should be incorporated in the SAAM proxy design. Rules for modification of existing rows elements and creation/deletion of rows have SMI-defined rules which are not apparent to neophytes. Operations allowed or disallowed by a routers SNMP agent are specified via variable access modifiers, the presence or absence of row-status or other table columns, all examples of which may be observed in a MIB browser or by analyzing the respective MIB source.

Recommended Follow-up Research:

A. Configure and Enable Ipv6 networking. This helps answer yet more questions required to further deploy SAAM into existing Cisco networks:

1. Ipv6 networking allows activation of more OID variables in the INT-SERV-MIB, particularly the IntservFlowId. The current test configuration would only return zero for that variable's value. My hypothesis is that going on to reconfigure the test subnet as an Ipv6 network and repeating the session and reservation configuration steps may be sufficient to allow non-zero flow Ids to appear in the IntservFlowID portion of the flow table. For more information, read "Configuring a Cisco 2611 for IPv4 and IPv6_ZD1011.RelA.pdf" on the report CD.
2. Cisco Ipv6 can run simultaneously with Ipv4. Interoperability issues may now be explored as a side issue. All Cisco QoS features which play a role in Flow and path-based routing must be uncovered, understood, and accounted for in any mechanism we arrive at for operating SAAM with Cisco networks, since a successful deployment must afford easy and as near fully automatic configuration and/or control of the routers by SAAM systems. Turning on Ipv6 also requires one of the latest versions of the IOS to be installed on the router: IOS v 12.2(T) has been mentioned in the literature as an official experimental release for this purpose. Experimentation with IOS v 12.2 (T) would also likely expose more QoS mechanisms not readily discussed in available literature, texts, or other nominal research sources.
3. IOS upgrade images are downloadable from the Cisco web site by parties with proper credentials (e.g. CCNA, CCIE) a valid service or vendors contract. Once obtained, images may be installed using tools included on the CD (tftp server).

B. Configure and enable Class-Based Weighted Fair Queuing, and other Cisco QoS- enabling queuing disciplines to further refine the SAAM-Proxy control model. See the following pdfs on the report CD:
Integrating Intserv and Diffserv in "rsvpscal.pdf" and RSVP Support for Low Latency Queuing in "rsvp_llq.pdf".

C. Move beyond the static RSVP test configuration commands and use dynamic signaling, reserving, and flow generation. This kind of experimentation is needed to study the Cisco QoS mechanisms under realistic operating conditions. Here's a ***link*** of one graduate student's work along these lines. Downloaded report and code are on the report CD. RSVP-enabled clients and servers of various types were used with Cisco routers. This type of work will lead to a more realistic testing and integration system as well.

D. D. Study as much as possible, all of Cisco's QoS features and mechanisms. There is no single document or procedure which ties all the Cisco QoS elements together which SAAM will need to consider if it is to effectively control a Cisco network. The required topics ranges go far beyond SNMP and mibs. Queuing, policy maps, access-lists, Ipv6 networking, RSVP, and much more should also be studied. They will need to be configured by someone… the ISP administrator who is deploying SAAM into his network, or a SAAM deployment representative, but it's best if as much as possible is performed automatically by the SAAM system integration units (Proxy boxes). After the initial installation, SAAM must then dynamically monitor and manage these attributes. The serial link connection between the SAAM proxy units and their Cisco router must serve for this type of privileged operation, as it assumes the highest form of trust: full physical and logical

security being granted the proxy units. The network connections between proxies and routers may provide higher speed SNMP/MIB transactions between the IOS-fixed router agent and the SAAM proxy units. However, security for network-based data passing will need to be researched. I do not believe that the MIB data are accessible for either reading or writing over the console serial port. See next figure.

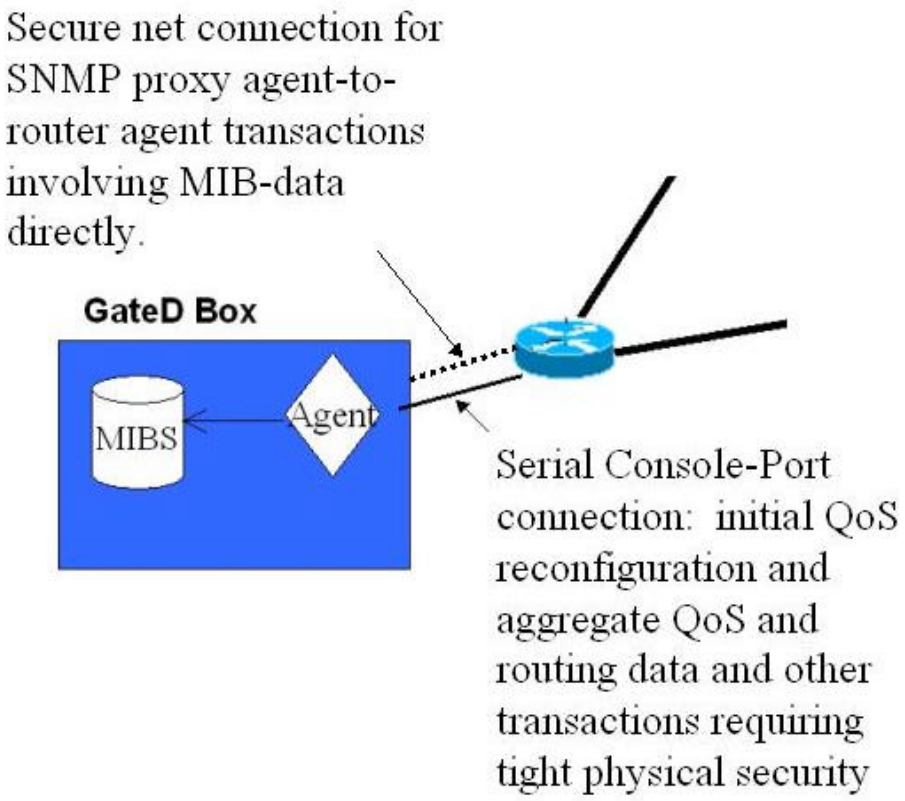

Figure 5. Privileged commands may be issued either over the network (e.g. un secure telnet, not SNMP) or via a serial connection to a router's console port. Network connections must be secure, thus telnet is out of the question. SSH (secure shell) login may not be secure enough. SNMP gets, puts, etc may not be accomplished over the serial connection. Thus, both types of connection may be required.

E. Investigate SNMP processing overhead implications to target routers (load and performance) and to SAAM processing effectiveness. SNMP is not a performance mechanism. Inappropriately frequent poling or large table walks and other operations must be performed in ways which avoid adversely impacting router forwarding and other operations…particularly on heavily used routers. Once all table objects SAAM will use are known, SNMP table walks or updates might be benchmarked and delays considered for their impact to SAAM effectiveness and responsiveness.

I’ve included many additionally documents on the report CD which may inspire further research along these lines.